

ALPHA
MICROSYSTEMS
RIGHT. FROM THE START.

ALPHA
MICROSYSTEMS
RIGHT. FROM THE START.

ALPHA
MICROSYSTEMS
RIGHT. FROM THE START.

ALPHA
MICROSYSTEMS
RIGHT. FROM THE START.

ALPHA
MICROSYSTEMS
RIGHT. FROM THE START.

ALPHA
MICROSYSTEMS
RIGHT. FROM THE START.

ALPHA
MICROSYSTEMS
RIGHT. FROM THE START.

ALPHA
MICROSYSTEMS
RIGHT. FROM THE START.

ALPHA
MICROSYSTEMS
RIGHT. FROM THE START.

ALPHA
MICROSYSTEMS
RIGHT. FROM THE START.

ALPHA
MICROSYSTEMS
RIGHT. FROM THE START.

ALPHA
MICROSYSTEMS
RIGHT. FROM THE START.

ALPHA
MICROSYSTEMS
RIGHT. FROM THE START.

AMOS[®] Command Files User's Manual

© 1995 Alpha Microsystems

REVISIONS INCORPORATED	
REVISION	DATE

00

December 1988

AMOS Command Files User's Manual

To re-order this document, request part number DS0-00071-00

The information contained in this manual is believed to be accurate and reliable. However, no responsibility for the accuracy, completeness or use of this information is assumed by Alpha Microsystems.

This document may contain references to products covered under U.S. Patent Number 4,530,048.

The following are registered trademarks of Alpha Microsystems, Santa Ana, CA 92799:

AMIGOS
AlphaBASIC
AlphaFORTRAN 77
AlphaMATE
AlphaWRITE
VIDEOTRAX

AMOS
AlphaCALC
AlphaLAN
AlphaNET
CASELODE

Alpha Micro
AlphaCOBOL
AlphaLEDGER
AlphaPASCAL
OmniBASIC

AlphaACCOUNTING
AlphaDDE
AlphaMAIL
AlphaRJE
VER-A-TEL

The following are trademarks of Alpha Microsystems, Santa Ana, CA 92799:

AlphaBASIC PLUS
DART
inFront/am

AlphaVUE
ESP

AM-PC
MULTI

AMTEC
inSight/am

All other copyrights and trademarks are the property of their respective holders.

ALPHA MICROSYSTEMS
2722 S. Fairview St.
P.O. Box 25059
Santa Ana, CA 92799

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION TO COMMAND FILES

1.1	WHAT IS A COMMAND FILE?	1-1
1.2	HOW DO YOU USE COMMAND FILES?	1-2
1.3	WHAT IS A DO FILE?	1-3
1.4	HOW DOES AMOS FIND COMMAND FILES?	1-3
1.5	GRAPHICS CONVENTIONS	1-4

CHAPTER 2 HOW TO CREATE COMMAND FILES

2.1	COMMENTS IN A COMMAND FILE	2-2
2.2	THE SYSTEM INITIALIZATION COMMAND FILE	2-2

CHAPTER 3 SPECIAL SYMBOLS AND COMMAND WORDS

3.1	BATCH	3-1
3.2	COM	3-1
3.3	CONT	3-2
3.4	EXIT	3-2
3.5	GOTO	3-2
3.6	PAUSE	3-3
3.7	:K	3-3
3.8	:P	3-3
3.9	:R	3-4
3.10	:S	3-4
3.11	:T	3-4
3.12	:U	3-4
3.13	:X	3-4
3.14	:< >	3-4
3.15	;	3-5
3.16	EXAMPLE	3-5

CHAPTER 4 THE IF STATEMENT

4.1	RESERVED WORDS USED WITH IF	4-1
4.2	FUNCTIONS USED WITH IF	4-2
	4.2.1 Examples	4-4
4.3	USING IF WITH SPECIAL DO SYMBOLS	4-4
4.4	CHECKING FOR ERRORS	4-5

CHAPTER 5 PASSING ARGUMENTS TO A COMMAND FILE

5.1	PARAMETER SYMBOLS	5-1
5.2	SPECIAL PARAMETER SYMBOLS	5-3
	5.2.1 \$D - Default Parameter List	5-3
	5.2.2 \$ - Null Parameter Symbol	5-3

CHAPTER 6 SPECIAL SYMBOLS

6.1	INFORMATION SYMBOLS	6-1
6.1.1	\$: - Original Device Symbol	6-1
6.1.2	\$P - Original Account Symbol	6-1
6.1.3	\$\$ - Real Dollar Sign	6-2
6.1.4	\$TM - Time of Day	6-2
6.1.5	\$TD - Current Date	6-2
6.1.6	\$TW - Day of the Week	6-2
6.1.7	\$NJ - Current Job	6-2
6.1.8	\$NT - Current Terminal	6-2
6.1.9	\$ND - Terminal Driver	6-2
6.1.10	\$NI - Interface Driver	6-3
6.1.11	\$NM - Modem Driver	6-3
6.1.12	\$NS - System Monitor	6-3
6.1.13	\$SV - Operating System Version	6-3
6.1.14	\$ND - Memory Available	6-3
6.1.15	\$UX - Radix	6-3
6.1.16	\$LG - Language	6-3
6.1.17	\$LY - Yes Symbol	6-3
6.1.18	\$LN - No Symbol	6-3
6.2	USER NAMES FEATURE SYMBOLS	6-4
6.2.1	\$NU - Current User Name	6-4
6.2.2	\$RP - Root Account	6-4
6.2.3	\$R: - Root Device	6-4
6.2.4	\$UB - User Privilege	6-4
6.2.5	\$UL - User Level	6-4
6.2.6	\$UE - User Expertise	6-4
6.3	HOW TO USE SPECIAL SYMBOLS	6-4

APPENDIX A SAMPLE COMMAND FILES

A.1	BACK.CMD	A-1
A.2	START.DO	A-2

GLOSSARY

DOCUMENT HISTORY

INDEX

CHAPTER 1

INTRODUCTION TO COMMAND FILES

One of the constants of everyday life is repetition. Especially in business, you find certain tasks have to be done on a regular basis. The mail has to be opened every day, somebody has to make the coffee, and if you don't do the payroll each week, people tend to get upset.

Some of these tasks are done on your computer. One of the great strengths of computers is their ability to free people from repetitious tasks, and to speed up the completion of those tasks. However, sometimes the operation of the computer itself can be repetitious.

Many times you do a certain task on the computer by using the same set of commands and operations every time. Alpha Micro provides special files called command and DO files that can contain these sets of commands and operations, making the ordinary operation of your computer easier.

This chapter discusses the following points:

- What is a command file?
- What is a DO file?
- How do you use command files?
- Graphics conventions

1.1 WHAT IS A COMMAND FILE?

One of the important features of the Alpha Micro system is that rather than having a set of "built in" commands AMOS recognizes, all of our commands are actually files stored on the disk.

When you enter an AMOS command (such as DIR), what you are really doing is telling AMOS to find and execute the file with that name. Because of this feature, you can extend and customize the set of AMOS commands on your computer by creating your own executable files. And, you can design your own commands, combining the existing AMOS commands with special instructions. You can also erase or rename standard AMOS commands if there are some commands you do NOT want people on your system to use, or if you want to control WHO knows and can use certain commands.

The way you build your own commands is by building a command file, which is a file that contains the same kinds of input you might enter from the keyboard. You can tell AMOS to read its instructions from a command file instead of entering those commands and data yourself.

1.2 HOW DO YOU USE COMMAND FILES?

Now that you know what a command file is, how do you use one? Let's take a small example—say you often erase all of your temporary files, then view the directory of your account. The process to do this is:

```
ERASE *.TMP RETURN
MEMO.TMP
ACCNTS.TMP
Total of 2 files deleted, 3 disk blocks freed

DIR RETURN
MEMO      TXT      12                DSK0:[1004]
ACCNTS    TXT       5
PAYROL    RUN       7
Total of 4 files in 34 blocks
```

This is a simple example—sometimes a frequently used sequence of commands may be quite long and tedious to type.

Let's say you create a command file called CLEAN.CMD to perform the functions in the example above. The file contains the following lines of text:

```
; This file erases TMP files and displays account directory
:T
ERASE *.TMP           ; Erase the temporary files
DIR                  ; Display the files in the account
```

The :T at the front of your command file is a special symbol allowing you to see the lines of your command file on your terminal as AMOS processes them. The semi-colon (;) indicates the text following it is a comment. All of these special symbols are discussed later in this manual. With this command file, instead of entering what you entered in the first example each time, you merely have to enter:

```
CLEAN RETURN
```

and AMOS reads the file CLEAN.CMD and performs the process for you. With this small example, you do not save much typing, but imagine the typing and time you would save if the sequence of commands is ten or twenty lines long!

If the extension of a command file is .CMD or .DO, you do not have to include the extension when entering the name of the command file (you saw you only had to type CLEAN rather than CLEAN.CMD).

Otherwise, you must specify the file extension; if your command file is called DOIT.TXT, you must include the .TXT when specifying the file. This is why it is easiest to use .CMD (or .DO) extensions.

1.3 WHAT IS A DO FILE?

A DO file is a command file that allows you to pass arguments to it from AMOS command level. You can input data, file names, and other information to the DO file when you execute the file. The special commands used in DO files are discussed in Chapters 5 and 6.

When you execute a DO file, all of the special symbols inside the file are immediately replaced with the values needed, then the file is executed just like a command file.

1.4 HOW DOES AMOS FIND COMMAND FILES?

When you enter a filename without an extension at AMOS command level, AMOS goes through the following procedure:












1. It looks in DSK0:[1,4] (SYS:) for a .LIT file with that name.
2. It looks in your current account for a .LIT file with that name.
3. It looks in DSK0:[1,4] (CMD:) for a .CMD file with that name.
4. It looks in DSK0:[1,4] (CMD:) for a .DO file with that name.
5. It looks in your current account for a .CMD file with that name.
6. It looks in your current account for a .DO file with that name.
7. It looks in your [X,0] account for a .LIT file with that name.
8. It looks in your [X,0] account for a .CMD file with that name.
9. It looks in your [X,0] account for a .DO file with that name.

You need to use a little care when selecting names for your command files so they don't conflict with any of the AMOS system commands (.LIT files) or any other command files on your computer.

1.5[∞]GRAPHICS CONVENTIONS

This manual conforms to the other Alpha Micro publications in its use of a standard set of graphics conventions. We hope these graphics simplify our examples and make them easier for you to use. Unless stated otherwise, all examples of commands are assumed to be entered at AMOS command level.

SYMBOL	MEANING
devn:	Device-Name. The "dev" is the three letter physical device code, and "n" is the logical unit number. Examples of device names are DSK0:, DSK5:, WIN1:, and MTU0:. Usually, device names indicate disk drives, but they can also refer to magnetic tape drives and video cassette recorders.
filespec	File Specification. A file specification identifies a specific file within an account. A complete filespec is made up of the devn:, the filename, the file extension, and the account number. For example: DSK0:SYSTEM.INI[1,4]
[p,pn]	This abbreviation represents an account on a disk you can store files and data in. An actual disk account number looks like this: [100,2] or [1,4]. Disk account specifications are sometimes referred to as "Project-programmer numbers."
{ }	Braces are used in some examples to indicate optional elements of a command line. In the example: DIR{/switch} the braces tell you "/switch" is not a required portion of the DIR command line.
/	The slash symbol precedes a command line switch or "option request." For example: DIR/WIDE:3 RETURN This command requests a directory display of the disk account you are currently logged into. The switch (/WIDE:3) indicates you want the display to be three columns wide.
TEXT	Bold text in an example of user/computer communication represents the characters you type.

SYMBOL	MEANING
TEXT	Text like this in an example of user/computer communication represents characters the computer displays on your terminal.
	In our examples, the keycap symbol appears whenever you need to press a certain key on your terminal keyboard. The name of the key you need to press appears inside the keycap symbol, like this:  . If you need to press the TAB key, you would see  , or the ESCAPE key,  . Sometimes the ESCAPE key is labeled ALT MODE.
 / 	This indicates a control sequence you press on the keyboard. The first key is pressed and held down while the second key is also pressed.
^	This symbol in front of a capital letter means the letter is a "control character." For example, when you press  /  , it appears on your screen as ^C. (^C is the control character that cancels most programs and returns you to AMOS command level.)
	This symbol means "halt!" It indicates an important note you should read carefully before going further in the documentation. Usually, text next to this symbol contains instructions for something you MUST or MUST NOT do, so read it carefully.
	This symbol means "hint." It indicates a helpful bit of information, or a "short cut" that could save you time or trouble.
	This symbol means "remember." It indicates something you should keep in mind while you are following a set of instructions.

CHAPTER 2

HOW TO CREATE COMMAND FILES

To create a command file, use AlphaVUE to make a text file (usually with a .CMD or .DO extension). Fill the file with the commands you would ordinarily enter from the keyboard, plus any special symbols. A command file can contain most commands or data you might enter from the keyboard; it can even contain the name of another command file.



If your computer uses the User Names feature, a User Name is required any time you log in from a logged off state. This does not usually affect command files unless the command file uses a LOGOFF command and then tries to log in again.

AMOS continues to read lines of text from the command file until the file ends. You may run, enter, and exit programs; supply data to programs, or perform system functions—all under the control of a single command file.



Do not use the MEMORY command inside a command file. This command changes the memory allocated to your job—the section of memory you have been assigned. Since your command file and any files it is using are in memory, changing memory may cause serious problems.

As an example, let's say you have a series of programs you need to run at the end of each week. You might create a command file like this:

```
:T
RUN PAYROL                      ; Runs the payroll for the week
RUN INVENT                      ; Updates inventory records
RUN COSTS                       ; Figures costs for the week
RUN SHIP                        ; Compiles report of shipping
RUN SUM                         ; Compiles weekly summary report
PRINT DSK3:[50,5]REPORT/C:4     ; Prints shipping report
PRINT DSK2:[156,2]SUMMARY/C:4   ; Prints summary report
SEND FRED REPORT IS PRINTING
```

The first line above is the special symbol that causes the lines of the command file to be displayed when the command file is executed. The next five lines run AlphaBASIC programs (the words after the semi-colon (;) indicate comments). The next two lines send files to a printer to be printed. Finally, the command file sends

user FRED a message the command file has finished its work.

2.1 COMMENTS IN A COMMAND FILE

You may include comments in your file by preceding them with a semicolon (as in the above example). AMOS does not process the comments. If a :T symbol is used, any comments that follow it are displayed on your terminal as the command file executes. If :R is used instead of :T, these comments are not seen, but will remain in the file to guide anyone who looks at the file.

Comments serve as explanations of the purpose and function of a command file. Often a command file will not be used by just one person, but by many. If a person wants to do a certain task, that person might look at the contents of a command file to see if it performs the needed task or not. If special symbols or unusual commands are in the file, the viewer might not know what they do, and so might not understand what the command file does.

Therefore, it is a good idea to fully comment your command files so other users will know exactly what functions the file performs.

2.2 THE SYSTEM INITIALIZATION COMMAND FILE

Of special interest to the System Operator is a unique command file, called the system initialization command file, the system uses every time you turn on or reset your computer.

This command file has special properties and commands that help AMOS tailor the system software for your particular hardware.

The System Operator can find information on the system initialization command file in the *System Operator's Guide to the System Initialization Command File*.

CHAPTER 3

SPECIAL SYMBOLS AND COMMAND WORDS

In addition to normal input you might make from your keyboard, there are special symbols and command words that appear only in command files.

The special symbols that begin with a colon (:) **MUST** appear at the start of a command file line in order to be read as special symbols. The comment character (;) and command words may be placed anywhere in the file. For examples of the use of these symbols, see Appendix A. The special symbols and commands are:

3.1 ° BATCH

BATCH loads frequently used AMOS system commands into your memory partition. You can use BATCH either at AMOS command level, or at the start of a command file. The commands it loads are: GOTO, IF, EXIT, TRACE, PAUSE and LOAD.

Use BATCH if you use many of these commands in your command files, since those commands execute faster if they are in your memory partition rather than on the disk.

3.2 ° COM

COM processes files based on their extensions. You use COM followed by a filespec (for a specific file—you cannot use wildcards). **DO NOT** include an extension for the filespec. Your command file will find the specified file and process it according to its extension.

If it is a .TXT file, it will format it using the TXTFMT command. If it is an AlphaBASIC source file (with a .BAS extension), it will compile it, etc. The file must be in your account number, but you may specify a different device.



Any commands in the command file after a COM statement will not be executed because COM does not return control to the command file after it finishes processing. Therefore use COM as the last statement in a command file.

3.3 CONT

This command, used either inside or outside (at AMOS command level) a command file, executes the command file CNT.CMD. This command continues the execution of a command file that was interrupted by the PAUSE command (see below).

3.4 EXIT

Terminates the processing of a command file. You use this in command files containing IF and GOTO statements. At the end of a conditional "branch," your command file must use a GOTO or an EXIT statement to go to another section or to end the file in order to avoid executing the next "branch." You may add a message after EXIT on the same command line that will be displayed on your terminal. The example shown in GOTO, below, also shows how EXIT works.

3.5 GOTO

Transfers control of a command file to a different line. The format is:

```
GOTO label
```

When the command file executes a GOTO line, it branches to the associated label in the command file and continues executing from there.

The label can be either a legal command file element, or it can be a comment. In either case, the label referred to must be on a line by itself, and must be after the GOTO statement in the context of the file. If the label is a comment, it must be preceded by a semi-colon with no intervening spaces, although the argument in the GOTO statement must not include the semi-colon. An example:

```
:R
IF LOOKUP("INVEN.DAT") = -1
GOTO OKAY
ELSE
:<The inventory file is missing!>
EXIT Thank you for using PROCES.CMD!
;OKAY
RUN INVENT
```



If you GOTO a command file element (such as :S or :<) the command file will begin execution AFTER that element without executing the element itself. Therefore it is better to use commented lines as labels.

3.6[∞]PAUSE

When PAUSE appears in a command file, the file stops and waits for you to input a character. If you press **RETURN**, the rest of the command file is then executed.

If you press any other key, the control file is suspended, and whatever is left in the command file is placed into a file called CNT.CMD. You may also put a message on the PAUSE line, which will display before the file pauses. For example:

```

.                               ; End of Daily Report
.                               ;
PAUSE Press RETURN to do a weekly report:
.                               ;
.                               ; Weekly Report section

```



Do not use the name "CNT.CMD" for your own command files, since PAUSE statements create a file named CNT.CMD, and write over any file of the same name that exists.

3.7[∞]:K

The :K symbol allows you to enter one line of data or commands to either AMOS or the program currently being executed.

When AMOS finds a :K symbol in your command file, it halts the processing of the file until you enter a line of characters that ends with **RETURN**. This input is then acted upon by AMOS or the program that is executing.

3.8[∞]:P

The :P symbol is used in conjunction with the :K symbol to let you enter part of a command line. Whatever you input is added to the line between the :P and :K symbols. For example, if you have the following command file:

```

:T
:<Input the name of the file
  you wish to VUE:
>
:P
VUE
:K

```

the command file will stop and wait for input, and then append to "VUE" whatever was entered, to make one command line. If TEST.TXT is entered, the command file will execute the command VUE TEST.TXT.

Put the :P before the command you want to add to, and do not forget the :K symbol after that line. Any other command file lines in between :P and :K will be appended together.

3.9[∞]:R

Causes the results of command file lines, but not the lines themselves, to be displayed. You can use multiple :S, :R, and :T symbols in one command file to let you see some program output and command file lines, but not others.

3.10[∞]:S

Suppresses the display of command file lines and program output generated by the command file. When :S is used, nothing the command file does will be displayed on the screen (except for messages). If you don't specify :R or :T, :S is assumed (the default).

3.11[∞]:T

Displays the lines of the command file and all of the output on your terminal as AMOS processes them.

3.12[∞]:U

Turns off the :T display. :R and :S will not override :T, so you must use :U to turn off :T before re-setting to :R or :S.

3.13[∞]:X

Ends the command file, even if the command file process is not at AMOS command level.

3.14[∞]:< >

These symbols allow you to display messages on your terminal as the command file executes. All the characters between the :< symbol and the > symbol are displayed.

A command file message is not acted upon by AMOS or any other program. A message may be more than one line long. The end of the message is indicated by the > symbol. If you forget the >, AMOS will print everything following the :< as a message, until it reaches the end of the file.

3.15[∞];

A semicolon marks a comment line (or part of a line) which is not processed, but is displayed with the rest of the command file if :T precedes the comment.

Certain commands, such as EXIT, allow you to put a message on the command line. If you put a comment after these commands, it will be taken as a message, and will display on the terminal, even if :R or :S is used.

3.16[∞]EXAMPLE

```
BATCH
:R
IF LOOKUP("INVEN.DAT") = -1          ; See if file exists
GOTO REPORT                          ; If so, go to REPORT label
ELSE
:< Inventory file was not found!  Report this to your
   supervisor before running again.
>
EXIT
;
;REPORT
;
RUN STOCK          ; Program to count stock
RUN DAILY          ; Program to compile daily report
;
PAUSE Press RETURN to produce a shortage list
;
:< If you want the CEO to get the list, enter CEO,
   otherwise enter the code for the reviewing manager:
>
:P
REPORT
:K
;
```


CHAPTER 4

THE IF STATEMENT

The IF statement allows you do different things in your command file depending on whether certain conditions are true or false. The basic format is:

```
IF expr
    List of commands to do when expr is TRUE
{ ELSE
    List of commands to do when expr is FALSE }
ENDIF
```

For example:

```
IF LOOKUP( "ACCT.TXT" ) = -1
    VUE ACCT.TXT
    EXIT
ELSE
    VUE ACCT.TXT
    Y
    EXIT
ENDIF
```

The ELSE clause is optional (if you don't use ELSE, the command file just goes past ENDIF if the condition is false) . IF statements can be nested up to 16 levels deep. Expressions can be made up of a wide variety of operators, reserved words, and functions, and support both string and floating point variables.

As much as possible, the expressions are compatible with AlphaBASIC. This makes it easier for people already familiar with the AlphaBASIC IF statement, and it makes it easier for AlphaBASIC programs to interact with command files.

4.1^{oo}RESERVED WORDS USED WITH IF

You can use a number of special reserved words with the IF statement:

ERROR	Tells you if an error occurred during the previous operation, and what kind of error it was. See the section below for the error reserved words and examples of how to use.
-------	---

SEVERITY	Tells you how severe the error that occurred in the last operation was. This code can be from 0 to 17 octal—the higher the number, the more severe the error.
TIME	The current time in string variable form. The time is in 24-hour format without seconds (for example, 1:00 PM is 1300).
DATE	The current date in a string variable of YYMMDD form (for example, January 15, 1987 is 870115).
TRUE	Equals TRUE for comparisons
FALSE	Equals FALSE for comparisons
CTRLC	Equals TRUE if <code>CTRL</code> / <code>C</code> was used at the keyboard

4.2 FUNCTIONS USED WITH IF

You can use the following functions with IF:

LOOKUP("n")	Equals -1 if the file specified inside the parenthesis is found, or 0 if it is not found. The filename must be enclosed in quotes, and is a standard file specification (if the file is in another account, include an account number, etc.).
DEVICE("n")	Returns one or more numbers describing the device if the device specified inside the parenthesis is found, or a 0 if it is not found. The device name must be enclosed in quotes. The returned values are:

1	File structured
2	Uses alternate track table
4	Can be divided into logical units
8	Sharable
16	Currently assigned to a job
32	Currently mounted
64	Set to "no network access"
128	Uses extended format directories
256	Uses paged bitmaps

DEVICE returns the total of all the applicable numbers above (i.e., a 48 would mean a device assigned to a job and mounted). These values are the flag values returned by the DEVCHR monitor call—see your *Monitor Calls Manual* for more information.

You can compare for an individual characteristic of the device, or any combination you wish. For example, if a device is file structured and can be divided into logical units, all of these statements would be true:

```

IF DEVICE( "DSK0:" ) = 1
IF DEVICE( "DSK0:" ) = 4
IF DEVICE( "DSK0:" ) = 5

```

NODE("n")	Equals -1 if the specified network node is available for access, or 0 if it is not. You must enclose the node number (or an equivalent ersatz name) in quotes.
BYTE(n)	Returns the value contained in memory byte n.
WORD(n)	Returns the value contained in memory word n.
LWORD(n)	Returns the value contained in memory longword n.
IO(n)	Returns the value contained in I/O byte n.

In addition, the following AlphaBASIC functions are supported:

ABS	ACS	ASC	ASN	ATN	CHR	COS	DATN
EXP	FACT	FIX	INSTR	INT	LCS	LEFT	LEN
LOG	LOG10	MID	RIGHT	RND	SIN	SGN	SPACE
SQR	STR	TAN	UCS	VAL			

Mathematical operators can also be used to evaluate expressions, in the same way you would use them in AlphaBASIC. The valid mathematical operators are:

+	addition, or unary plus
-	subtraction, or unary minus
*	multiplication
/	division
^	raise to power
**	raise to power
NOT	logical NOT
AND	logical AND
OR	logical OR
XOR	logical XOR
EQV	logical equivalence
MIN	minimum value
MAX	maximum value

You may also use the following relational operators:

=	equal
<	less than
>	greater than
<>	unequal
#	unequal
<=	less than or equal
>=	greater than or equal

IF also fully supports AlphaBASIC substring modifiers (use of the [and] symbols and number to designate parts of string variables). See your *AlphaBASIC User's Manual* for more information.

All operators are evaluated in the same order as in AlphaBASIC—of course, you can use parentheses to override the usual order.

4.2.1[∞]Examples

Let's take a few examples to see how the above reserved words and operators can be used. Say you want your command file to check to see if a certain file exists before you run an AlphaBASIC program that writes to that file. The lines in your command file might be:

```
IF LOOKUP("INVEN.DAT") = -1
  RUN STOCK
ELSE
  :< INVENTORY FILE NOT FOUND! >
  EXIT
ENDIF
```

Another example:

```
IF TIME >= "1200"
  :<Good afternoon!
  >
ELSE
  :<Good morning!
  >
ENDIF
```

As you can see, the IF program allows you to do many things within command files. It gives your command files a great deal of flexibility and control over various situations.

4.3[∞]USING IF WITH SPECIAL DO SYMBOLS

Chapter Six contains a list of special DO file symbols. When using any of these symbols that return a string value with the IF statement, you must use quotes. For example:

```
IF "$NU" = "John Smith"
  :< Good day, Johnny!
  >
ELSE
  :< Hello.
  >
```

4.4[∞]CHECKING FOR ERRORS

The following reserved words are defined so you can check for specific errors and conditions reported by the monitor:

STANDARD ERROR CODE	RESERVED WORD	ERROR
0		No error detected
1	ERR'SPC	File specification error
2	ERR'MEM	Insufficient free memory
3	ERR'FNF	File not found
4	ERR'FAX	File already exists
5	ERR'RDY	Device not ready
6	ERR'FUL	Device full
7	ERR'ERR	Device error
8	ERR'USE	Device in use
9	ERR'ILC	Illegal user code
10	ERR'PRV	Protection violation
11	ERR'WRT	Write protected
12	ERR'TYP	File type mismatch
13	ERR'DNX	Device does not exist
14	ERR'IBN	Illegal block number
15	ERR'INI	Buffer not INITed
16	ERR'FNO	File not open
17	ERR'FAO	File already open
18	ERR'KPT	Bitmap kaput
19	ERR'MNT	Device not mounted
20	ERR'IFL	Invalid filename
21	ERR'BBH	BADBLK.SYS has a bad hash total
22	ERR'BBW	BADBLK.SYS is in unsupported format
23	ERR'BBN	BADBLK.SYS not found
24	ERR'NOQ	Insufficient queue blocks
25	ERR'MFD	MFD is damaged
26	ERR'LMN	First logical unit is not mounted
27	ERR'RNR	Remote is not responding
28	ERR'FIU	File in use
29	ERR'RIU	Record in use
30	ERR'EMB	Deadly embrace possible
31	ERR'DEL	File cannot be deleted
32	ERR'REN	File cannot be renamed
33	ERR'RNL	Record not locked
34	ERR'RNO	Record not locked for output
35	ERR'LQF	LOKSER queue is full
36	ERR'NFS	Device is not file structured
37	ERR'IRS	Illegal record size
38	ERR'BLK	Block allocate/deallocate error
256	ERR'MSC	Miscellaneous error
257	ERR'MAP	Memory map destroyed
258	ERR'IPR	Insufficient privileges to run program

STANDARD ERROR CODE	RESERVED WORD	ERROR
259	ERR'L12	Must be logged into [1,2]
260	ERR'OPR	Must be logged into DSK0:[1,2]
261	ERR'M20	Program requires 68020 processor
262	ERR'LOG	Must be logged in to run program
263	ERR'BER	Bus error
264	ERR'PAR	Memory parity error
265	ERR'ADR	Address error
266	ERR'IIN	Illegal instruction
267	ERR'DIV	Divide by zero error
268	ERR'CHK	CHK instruction trap
269	ERR'TRP	TRAPV instruction trap
270	ERR'PRI	Privilege violation
271	ERR'TRC	Trace trap return
272	ERR'EM1	EM1111 instruction trap
273	ERR'MEX	Miscellaneous exceptions
274	ERR'II0	Illegal interrupt on level 0
275	ERR'II1	Illegal interrupt on level 1
276	ERR'II2	Illegal interrupt on level 2
277	ERR'II3	Illegal interrupt on level 3
278	ERR'II4	Illegal interrupt on level 4
279	ERR'II5	Illegal interrupt on level 5
280	ERR'II6	Illegal interrupt on level 6
281	ERR'II7	Illegal interrupt on level 7
282	ERR'BTO	Bus time-out error
283	ERR'MMU	MMU error
284	ERR'PPV	Co-processor protocol violation
285	ERR'FUB	FPCP branch or set on unordered condition
286	ERR'FIR	FPCP inexact result
287	ERR'FDZ	FPCP divide by zero
288	ERR'FUN	FPCP underflow
289	ERR'FOE	FPCP operand error
290	ERR'FOV	FPCP overflow
291	ERR'FSN	FPCP signaling NAN
292	ERR'MCE	MMU co-figuration error
293	ERR'MIO	MMU illegal operation
294	ERR'MLV	MMU access level violation
384	ERR'LPA	Language processor aborted
385	ERR'LAE	Language processor aborted with errors
386	ERR'LCE	Language processor completed with errors
387	ERR'UND	Undefined identifier in input
388	ERR'RIE	Runtime interpreter error
389	ERR'SYN	Syntax error in input
390	ERR'AEL	Assembly error in linkage
512	ERR'CTC	Process aborted by operator (^C)

STANDARD ERROR CODE	RESERVED WORD	ERROR
513	ERR'CLF	Command line format error
514	ERR'CLS	Command line switch error
515	ERR'SSD	Bad SSD
516	ERR'ONF	Overlay not found
0	SEV'NUL	Operation completed - no errors/warnings
2	SEV'WRN	Operation completed with warnings only
4	SEV'FER	Operation completed with errors
6	SEV'AER	Operation aborted due to errors or ^C
8	SEV'FTL	Operation aborted due to fatal error

Using the above reserved words, you can check for problems that might occur when you run your command file. For example:

```
RUN STOCK
IF ERROR <> SEV'NUL
:< ERROR DETECTED IN STOCK PROGRAM! >
EXIT
ENDIF
```

or:

```
IF ERROR = ERR'L12
LOG 1,2
ENDIF
```

CHAPTER 5

PASSING ARGUMENTS TO A COMMAND FILE

Another versatile feature of command files is, by giving the command file a .DO extension you can pass arguments to the file when you "run" it. When you enter the name of the command file at AMOS command level, you may also include one or more arguments. For example:

```
FORMAT MEMO RETURN
```

in this case, "MEMO" is the name of a file you want your FORMAT command file to work on. If your command file takes more than one argument, separate them with spaces. For example:

```
FORMAT MEMO TAX.DAT RETURN
```

If an argument contains two or more words, or spaces, you can enclose the argument in <> symbols. For example:

```
UPDATE MEMO <DATA FOR THURSDAY> RETURN  
DIAL <1 456 384 8392>
```

5.1°PARAMETER SYMBOLS

Each argument passed to a DO file has an associated parameter symbol. These consist of a dollar sign (\$) and a number. The first argument passed to a command file is \$0, the second is \$1, and so on, up to \$9. This parameter symbol is like a variable—it represents the argument. Let's take an example to see how this works. We used the command line:

```
FORMAT MEMO TAX.DAT RETURN
```

above. The FORMAT.DO file might look like this:

```
:R  
TXTFMT HEADER.TXT,$0,$1  
:< TEXT FORMATTING COMPLETED  
>
```


When the DO file is called, the \$0 and \$1 symbols are replaced with the arguments specified at AMOS command level (in this case, MEMO and TAX.DAT).

Argument list items are associated with parameter symbols—NOT in the order the parameter symbols appear. The first item in the argument list is associated with parameter \$0, even if parameter \$2 appears before \$0 in the command file. Here is another example:

```
:R
DIR/W $2
TXTFMT HEADER,$0
RENAME/D $0.LST=HEADER.LST

TXTFMT HEADER,$1
RENAME/D $1.LST=HEADER.LST
PRINT $0,$1
PRINT
```

If you name this file DOC.DO and run it, entering:

```
DOC PSTINV ACTPAY *.TXT RETURN
```

the \$0 is replaced with PSTINV, the \$1 is replaced with ACTPAY, and the \$2 is replaced with *.TXT. Thus, the DO file above will be executed as if it were:

```
:R
DIR/W *.TXT
TXTFMT HEADER,PSTINV
RENAME/D PSTINV.LST=HEADER.LST

TXTFMT HEADER,ACTPAY
RENAME/D ACTPAY.LST=HEADER.LST
PRINT PSTINV,ACTPAY
PRINT
```

You can use a parameter symbol to represent an entire file specification, a portion of a file specification, a command, or any other piece of text inside a command file.



Remember the value of a parameter is set at the time of execution of a DO file—with some parameters, this can sometimes cause different results than you might expect. As an example, look at this DO file:

```
:R
sleep 14400
:<$TM>           ; $TM returns the time of day
```

The \$TM is replaced by the current time. However, since the DO file's first instruction is to sleep for four hours, the time printed on the screen will be four hours early. If you run the DO file at noon, at four o'clock the

DO file will tell you it is noon.

If you have more items in your argument list than there are parameter symbols in the command file, the extra items are ignored. If you have fewer items in your argument list than there are parameter symbols, the extra parameters are ignored.

If you do not supply a parameter, however, the line it appears on may still be processed, especially if there is a default for the specified command. For example, if a line in your command file is DIR \$0, and you do not specify that parameter, AMOS will assume the default of DIR *.* , and process the line in that manner.

5.2 SPECIAL PARAMETER SYMBOLS

In addition to the usual command file parameter symbols and the special command file symbols (discussed in Chapter 6), these special parameter symbols allow you to use DO files in a more flexible way and for a greater range of applications:

5.2.1 \$D - Default Parameter List

If you specify fewer items in the argument list than there are parameter symbols in the command file, AMOS usually ignores the extra parameter symbols. You can, however, supply a default argument list that AMOS will use if you omit all or part of the argument list.

On the first line of your DO file, enter \$D followed by a list of defaults, in order. For example, say you do most of your work in one file, called PAYROL.DAT. You can create a DO file to save yourself typing. It might be called LOOK.DO and look like this:

```
$D PAYROL.DAT
:R
VUE $0
```

If you enter just LOOK^{RETURN} at AMOS level, you get PAYROL.DAT—if you want to VUE some other file, you add the filename after LOOK. PAYROL.DAT becomes the default for \$0.



If a \$D line appears in your command file, it MUST be the first line of the file (even before a :T, :S, or :R symbol).

5.2.2 \$ - Null Parameter Symbol

A single \$ indicates a null parameter in a default parameter list, or a null argument in an argument list. This symbol allows you to designate which parameter will be associated with which argument. For example:

```
$D $ INVEN.TXT
:R
TXTFMT HEADER.TXT,$0
RENAME $0.LST=HEADER.LST
VUE $1
```

In the file above, the \$ after \$D indicates there is no default for \$0, and INVEN.TXT is the default for \$1.

CHAPTER 6

SPECIAL SYMBOLS

The symbols below can be used with the IF statement to make tests within your DO files, and sometimes they can be used to display information when you run the command file.



The symbols in this chapter only work with files with .DO extensions.

6.1 INFORMATION SYMBOLS

The symbols in this section return information your DO file can use in IF conditions to customize your operations.

6.1.1 $\$$ - Original Device Symbol

Represents the device the DO file was logged into when the file was run. For example, if you are logged into an account on DSK0:, the line:

```
LOG $:[1,4]
```

is executed as:

```
LOG DSK0:[1,4]
```

You can use this symbol in combination with the original account symbol (\$P) to keep track of the account and device where the command file originated. This is useful if the command file logs into other accounts, and you want to return to the starting point when done.

6.1.2 $\$P$ - Original Account Symbol

Represents the account the DO file was logged into when the file was run. For example, if you are logged into account [230,5], the command file line:

```
DIR DSK0:[ $P]
```

is executed as:

```
DIR DSK0:[230,5]
```

6.1.3[∞]\$\$ - Real Dollar Sign

Represents an actual dollar sign. If you need to use a dollar sign as part of a message within a command file, you enter it as two dollar signs to distinguish it from the null parameter symbol (see Chapter 5).

6.1.4[∞]\$TM - Time of Day

The current time of day as a string in 24-hour format, including separator, without seconds. For example, "13:00."

6.1.5[∞]\$TD - Current Date

The current date as a string in "YYMMDD" format. For example, "870719" for 19 August 1989.

6.1.6[∞]\$TW - Day of the Week

The current day of the week returned as a number, starting with 0 for Monday, 1 for Tuesday, etc.

6.1.7[∞]\$NJ - Current Job

The name of the current job as a string.

6.1.8[∞]\$NT - Current Terminal

The name of the currently attached terminal as a string.

6.1.9[∞]\$ND - Terminal Driver

The name of the terminal driver currently in use, as a string.

6.1.10[∞]\$NI - Interface Driver

The name of the interface driver currently in use, as a string.

6.1.11[∞]\$NM - Modem Driver

The name of the modem driver currently in use, as a string.

6.1.12[∞]\$NS - System Monitor

The name of the system monitor as a string. For example, "AMOS32."

6.1.13[∞]\$SV - Operating System Version

The current operating system version as a string.

6.1.14[∞]\$ND - Memory Available

The number of bytes of memory currently assigned to the user's job.

6.1.15[∞]\$UX - Radix

Gives you the currently active radix (octal or hexadecimal) returned as a number (8 or 16). This is the way the system is currently storing and using numbers. AMOS usually uses octal representation. The SET program can be used to change the Radix.

6.1.16[∞]\$LG - Language

The name of the currently selected language as a string. For example, "ENGLISH."

6.1.17[∞]\$LY - Yes Symbol

The single character that stands for "yes" in the currently selected language, as a string.

6.1.18[∞]\$LN - No Symbol

The single character that stands for "no" in the currently selected language, as a string.

6.2[∞]USER NAMES FEATURE SYMBOLS

The special symbols in this section have to do with the user names feature, which allows programs to be tailored to the specific abilities of the user. See your *System Operator's Guide* for more information about User Names.

6.2.1[∞]\$NU - Current User Name

The user name of the current user as a string.

6.2.2[∞]\$RP - Root Account

The current user's root account number as a string. For example, "120,0."

6.2.3[∞]\$R: - Root Device

The current user's root device as a string. For example, "DSK0:".

6.2.4[∞]\$UB - User Privilege

The current user's privilege, returned as a decimal number. For example, "10."

6.2.5[∞]\$UL - User Level

The current user's level (0 to 100).

6.2.6[∞]\$UE - User Expertise

The current user's expertise level (0 to 100).

6.3[∞]HOW TO USE SPECIAL SYMBOLS

You can use special symbols to display information to the user of your command file. You can place them anywhere within the :< and > symbols, and the symbol will be replaced by the appropriate information. For example:

```
:R
:<Good Morning, $NU.  You are running $NS $SV.
>
```

When run, the above command file will print something like this:

```
Good Morning, John Smith.  You are running AMOS/L 2.0A(175).
```

You can also use special symbols with other command file commands. For example:

```
:R
IF "$NJ" <> "FRED"
:<Welcome to Fred's account.
>
ENDIF
```

Here is an example of a more complex command file that makes use of many of the special symbols:

```
:R
XY = 0
LOG $R:$RP
:<Good >
IF TIME < 1200
:<Morning,>
ELSE
:<Afternoon,>
ENDIF
:< $NU! It is $TM. You are job: $NJ,
working on terminal $NT, using terminal driver
$ND with the $NI interface. The system
monitor is $NS. You are operating under version
$SV. Your root account is $R:$RP. Your
privileges are $UB, and your level is $UL, with
an expertise level of $UE. There are $ND bytes of
memory for your job.
>
```

The display of this command file would look something like this:

```
LOG DSK2:[100,1]
Logged into DSK2:[100,1]
Good Afternoon, John Smith! It is 13:00. You are job:
JOHN, working on terminal SMITH, using terminal driver
ALPHA with the AM350 interface. The system
monitor is AMOS/L. You are operating under version
2.0A(175)-2. Your root account is DSK2:[100,1]. Your
privileges are 2149515264, and your level is 50, with
an expertise level of 45. There are 371573 bytes of
memory for your job.
```


APPENDIX A

SAMPLE COMMAND FILES

Below are some examples of the kinds of command files you can create to help you perform frequently used sequences of commands. If you want all of the users on the system to be able to share your command files, have the System Operator copy them into the System Command File Library, DSK0:[2,2] (CMD:). If a command file is in this account, it can be run by any user on the system without any disk or account specifications.

A.1 BACK.CMD

This command file transfers backup copies of a disk to a VCR tape.

```
:T
:< Backs up the entire system >
KILL MALSER ; Turn off AlphaMAIL job
WAIT MALSER
:< Alert other jobs of backup >
SEND * <PERFORMING BACKUP... SYSTEM COMING DOWN.
SLEEP 25
LOG DSK0:1,2
ERASE *.LST
DSKANA DSK3:DSK0=DSK0:/E
DSKANA DSK3:DSK1=DSK1:/E ; Run DSKANA on disks
DSKANA DSK3:DSK2=DSK2:/E
SET NOLINK 16842754- ; Turn off AlphaNET
DIRSEQ DSK0:[ ]/E
DIRSEQ DSK1:[ ]/E ; Sequence accounts
DIRSEQ DSK2:[ ]/E
ERASE DSK0:[ ]*.BAK,DSK1:[ ]*.BAK,DSK2:[ ]*.BAK
BACKUP/C:5 DSK0:[ ],DSK1:[ ],DSK2:[ ]
CHR. SALES
JAN 87
TREE DIV.
Big Business ; Answer BACKUP's
Accounting ; questions
O. Hardy
; Certify the
CRT610/F ; backup was good
```

A.2[∞]START.DO

AMOS automatically looks for a START.CMD or START.DO file when you log into an account. If it finds one, it executes it. You can use a START file to customize accounts, setting up features of AMOS specifically for the needs of the person using the account. Here's an example:

```
:S
LOAD DSK0:[1,4]AMSORT.SYS
SET PROMPT Accounts Payable
:R
DIR/W
DEL *.PFK
:<
>
XY = 32                                ; Set highlight text
IF TIME > 1200 THEN
:< Good Afternoon,>
ELSE
:< Good Morning,>
ENDIF
;
IF "$NU" = "Carl Anderson"
LOAD DSK2:[100,0]MY.PFK                ; Load personal function
:< Carl. >                             ; key table
XY = 33                                ; Turn off highlight
:<
>
ELSE
IF "$NU" = "Fred Sheldon"
:< Fred. >
XY = 33                                ; Turn off highlight
LOAD DSK1:FRED.PFK[60,0]              ; Load Fred's function table
:<
>
ELSE
:< $NU. Instructions for working with the accounts payable
    files are in HELP.TXT. >
XY = 33                                ; Turn off highlight
:<
>
ENDIF
ENDIF
```

INDEX

SYMBOLS

\$	5-4
\$\$	6-2
\$:	6-1
\$D	5-3
\$LG	6-3
\$LN	6-3
\$LY	6-3
\$ND	6-3
\$NI	6-3
\$NJ	6-2
\$NM	6-3
\$NS	6-3
\$NT	6-2
\$NU	6-4
\$P	6-1
\$R:	6-4
\$SV	6-3
\$TD	6-2
\$TM	6-2
\$TW	6-2
\$UB	6-4
\$UE	6-4
\$UL	6-4
\$UX	6-3

: (colon)	3-1
:<>	3-4
:K	3-3
:P	3-3
:R	3-4
:S	3-4
:T	3-4
:U	3-4
:X	3-4
;	3-5

A

AlphaBASIC functions	4-3
AMOS commands	1-1
Arguments	5-1

B

BYTE	4-3
------	-----

C

Colon	3-1
COM	3-1
Command files	1-1
Comments	2-2
Creating	2-1
Messages	3-4
Samples	A-1
Use	1-2
Comments	2-2
CONT	3-2
Control-characters	
CTRL / C	1-5
CTRLC	4-2

D

DATE	4-2
DEVICE	4-2
DO files	1-3, 5-1
Parameter symbols	5-1, 5-3

E

ELSE	4-1
ENDIF	4-1
ERROR	4-1
Error checking	4-5
Error codes	4-5
EXIT	3-2

F

FALSE	4-1 to 4-2
Functions	4-2

G

GOTO	3-2
Graphics conventions	1-4

I

IF	4-1
AlphaBASIC functions	4-3
Error checking	4-5
Examples	4-4
Functions	4-3
Mathematical operators	4-3
Relational operators	4-3
Reserved words	4-2
IO	4-3

L		
	LOG	2-1
	LOOKUP	4-2
	LWORD	4-3
M		
	Mathematical operators	4-3
N		
	NODE	4-3
P		
	Parameter symbols	5-1, 5-3
	Passing arguments to command files	5-1
	PAUSE	3-3
R		
	Relational operators	4-3
	Reserved words	4-2
S		
	Sample command files	A-1
	SEVERITY	4-2
	Special symbols	3-1, 6-1
	Use	6-4
	Special words	3-1
	System initialization command file	2-2
T		
	TIME	4-2
	TRUE	4-1 to 4-2
U		
	User names feature	2-1
W		
	What a command file is	1-1
	What a DO file is	1-3
	WORD	4-3